Problem 8 Editorial: your a wizard harry
6+6=12 Points

Problem ID: `tower`
Rank: `2+4`

Overview

Problem statement: https://calico.berkeley.edu/files/calico-fa22/contest/tower/tower.pdf

Fun fact: although the problem statement itself contains many rhyming terms and vaguely skirts the border of an internationally-renowned intellectual property, the problem premise itself went through many iterations before arriving at the final result! From a lumberjack with an axe of growing length, to a powerdrill of increasing power, to a brief stint with pogo sticks and jump height, this problem has really seen it all. A main challenge in writing the problem was in juggling the use of many terms while aiming to keep the mechanics relatively simple.

The main test set allowed for a complete search via a direct simulation of the problem; its forgiving constraints were geared towards making the problem accessible to beginners.

The bonus test set required optimizations to be made on the brute-force approach, lending itself to a computationally-centric solution involving a skip list-esque data structure and an approach that mirrors that taken by binary exponentiation or decomposition techniques such as square root decomposition!

Another contestant, Austin Geng from team OlyFans, found a mathematical solution during the contest that lends itself to a nice visual representation—the explanation of which is featured below. Not only is its implementation very concise (and contributed by Austin!), but it pieces together simple concepts with a core observation to create a solution that runs faster than the solution we officially planned to release! We highly encourage you to read about it below, it's very enlightening.

Main Test Set

Try 'em All *Solutions available in Java, Python*

As the name of the approach suggests, we consider using each of the $\mathbf{N}$ towers as a starting point to calculate the time needed to destroy all other towers. We then take the minimum of these times to get our answer.

It's only necessary to keep track of the elapsed time, since it is conveniently always equal to Herry's current power. At each tower, accumulate the time Herry spends waiting before he is able to destroy the tower—equal to the difference between Herry's current power and the power of the tower, or zero if Herry's power at least matches that of the tower. Then, accumulate the time spent walking between towers.

Some things to keep in mind are that Herry begins at each starting tower with a power of zero, and must destroy it before continuing to the other towers. Additionally, there will be one stretch of distance between the first and last towers that Herry doesn't need to travel, since we only require the time needed to destroy all the towers.

As we need to consider $\mathbf{N}$ starting points, with each simulation taking $O(\mathbf{N})$ time to carry out, our overall runtime is $O(\mathbf{N}^2)$.

Bonus Test Set

With 10^5 towers, quadratic time is out of the question. We need something much faster.

Decomposing into Twos *Solutions available in Java, Python*

The main motivation here is that we still want to find the time needed to destroy all towers from each of $\mathbf{N}$ starting points. However, we notice that while carrying out each of the $\mathbf{N}$ simulations, we end up performing a lot of redundant work. Since we're visiting the towers in the same relative order and traversing the same edges multiple times, there seems to be an opportunity for us to build on previously-done calculations to save ourselves time.

The desire to save work can be motivated by the observation that at any point in time, once Herry has accumulated a high enough power, he will no longer have to wait at any future towers—one such value can be the highest power of any tower. We can generalize this idea to make another observation: if Herry arrives at a tower while possessing more power than it, his surplus of power can possibly remove the need for Herry to wait at other towers in the future.

Let's consider two scenarios:

1. If Herry, with a power of p, arrives at tower i with a higher power $\mathbf{P_i} > p$, Herry must wait until his power equals that of the tower to destroy it (specifically, $\mathbf{P_i} - p$ hours). Afterwards, Herry will have a power equalling that of the destroyed tower ($p = \mathbf{P_i}$).
2. However, if Herry arrives at a tower with a $\mathbf{P_i} \leq p$, he can instantly destroy the tower. Afterwards, Herry's power remains unchanged. This power can be thought of as the minimum power required to destroy the tower, plus an additional offset q ($p = \mathbf{P_i} + q$).

If, in the first scenario, Herry later comes across another tower j with a higher power $\mathbf{P_j}$ than him, Herry will have to wait $\mathbf{P_j} - p$ hours, as usual. However, if Herry comes across the same tower j with power $\mathbf{P_j}$ following the second scenario, Herry can wait up to q hours less! Noticeably, however, this discount only applies to time spent waiting at towers—time spent walking is nonnegotiable.

We can use this information to introduce the idea of an arc: an arc from $i \rightarrow j$ will represent the time required to destroy all towers starting from tower i until tower j, assuming we've just destroyed tower i. The goal here is to find a way to combine arcs that is independent of how many towers exist between towers i and j. If possible, we could quickly calculate the time needed to destroy large amounts of towers without large amounts of redundant calculations.

The main challenge with combining arcs is that, depending on where Herry starts, he may have different powers upon arriving at the same tower. However, the only information that is relevant to each arc is Herry's power surplus after destroying the final tower. As mentioned above, this surplus can be used to remove time Herry would've spent waiting at towers in the future. Each $i \to j$ arc will therefore have a walking time $w_{i,j}$ and stopping time $s_{i,j}$, denoting the time spent walking and time spent stopped at towers throughout this arc, respectively. Together, they sum to the total time needed to traverse the arc.

We can merge two arbitrary arcs $i \to j$ and $j \to k$ into arc $i \to k$ as follows:

1. Calculate Herry's power after destroying tower j: $p = \mathbf{P_i} + s_{i,j} + w_{i,j}$
2. Offset the $j \to k$ arc's stop time $s_{j,k}$ with Herry's surplus of power: $s_{j,k} = \max(0, s_{j,k} - (\mathbf{P_j} - p))$
3. $s_{i,k} = s_{i,j} + s_{j,k}$ and $w_{i,k} = w_{i,j} + w_{j,k}$

Each arc starts with the assumption that the first tower has already been destroyed to better facilitate the combination of arcs. With this process, we can combine pre-existing arcs to get new arcs of doubling lengths. As such, we can find all arcs of length 2, 4, 8, and so on, until the arc lengths we achieve are the largest power of 2 that doesn't exceed $\mathbf{N}$. There are $O(\log\mathbf{N})$ steps in this process, each of which takes $O(\mathbf{N})$ time—meaning this step takes $O(\mathbf{N}\log\mathbf{N})$ time.

Ultimately, we want to find all arcs of length $\mathbf{N}$; these would provide us with the same values as our previous solution mentioned above. However, since $\mathbf{N}$ may not be a power of two itself—we have to build these arcs ourselves. This can be done by greedily appending the next longest arc that doesn't take you past your destination, until your arc is $\mathbf{N}$ long. For example, an arc from towers $1 \to 7$ (7 long) can be built from arcs $1 \to 4$, $4 \to 6$, and $6 \to 7$. A path of length $\mathbf{N}$ will use at most $\log\mathbf{N}$ of these arcs, due to the remaining distance being at least halved at each step—meaning the time needed to "simulate" the time needed to complete all $\mathbf{N}$ arcs (in other words, the time needed to destroy all towers from each of the $\mathbf{N}$ possible starting towers) is now $O(\mathbf{N}\log\mathbf{N})$.

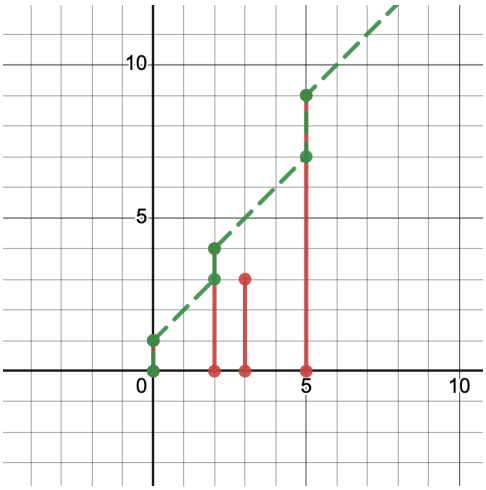
Those more familiar with algorithms may draw connections between this approach and other common decomposition techniques, such as square root decomposition. When building our arcs of length $\mathbf{N}$, the technique used parallels that of binary exponentiation or skip lists.

Wait a Minute *Solution available in C++ (solution provided by Austin Geng)*

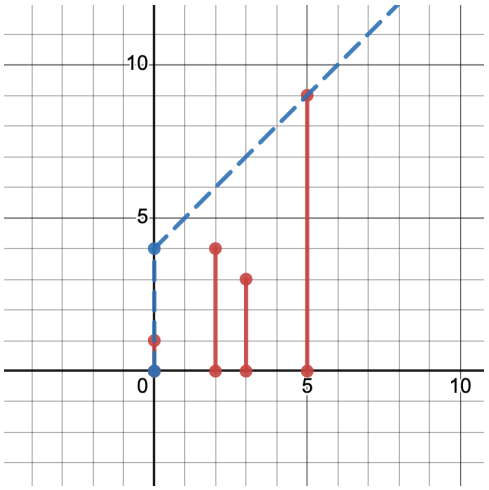
Special thanks to Austin Geng of team OlyFans for sharing this approach with us and allowing us to use his code.

This solution hinges on the observation that it is acceptable for Herry to do all the waiting upfront, followed by continuously walking around the circle—destroying towers instantly when he comes across them. Assuming Herry waits the minimum amount of time required before walking around the circle, this approach takes just as long as greedily walking forward whenever possible.

We can visualize an example using $\mathbf{N}$ = 4 towers with powers $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$, $\mathbf{P_4}$ = 1, 4, 3, 9 (the red line segments) and distances $\mathbf{D_1}$, $\mathbf{D_2}$, $\mathbf{D_3}$, $\mathbf{D_4}$ = 2, 1, 2, 2 ($\mathbf{D_4}$ is not explicitly visible, due to it wrapping around from tower #4 back to tower #1). Say Herry chooses to start at tower #1. The x-axis represents distance along the circle, and the y-axis represents Herry's power:



Greedily moving as much as possible, stopping and waiting when unable to immediately destroy a tower



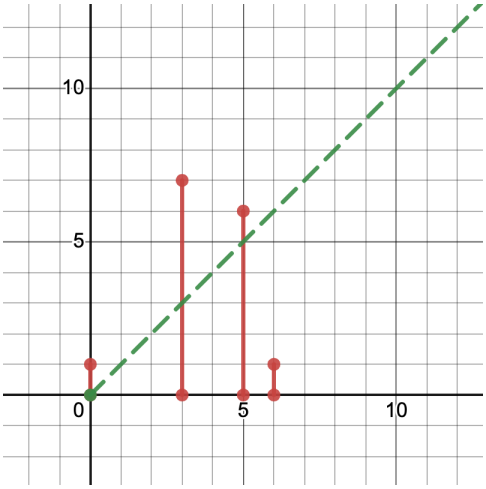
Waiting the minimum time before continuously moving, instantly destroying towers upon arrival

By walking forward greedily, Herry spends an hour waiting at tower #1, another hour waiting at tower #2, and two hours waiting at tower #4. Herry spends an additional 5 hours walking, meaning the towers are destroyed in a total of 9 hours. If, instead, Herry waited 4 hours at the beginning before walking continuously for 5 hours, he could also destroy the towers in 9 hours. Herry will never spend additional time waiting at towers—while walking, his power is always at least equal to his power at the same x-distance during the greedy approach.

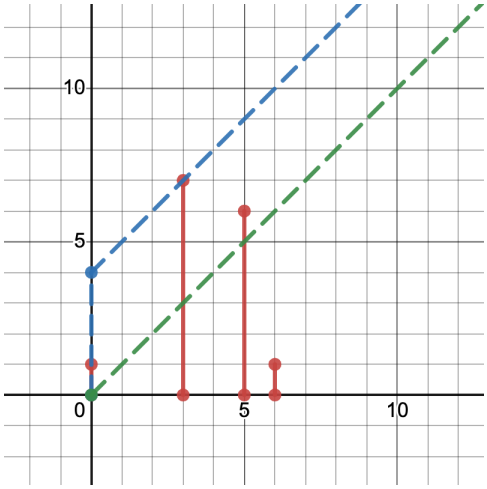
We can leverage this observation to efficiently determine how long Herry needs to destroy all towers from any given starting point. The approach consists of two phases:

1. Determine the time needed to destroy all towers from an arbitrary starting point, such as tower #1.
2. Take a “step back,” making the previous tower our new starting point (and pushing all other towers “further back” from Herry’s perspective); calculate the time needed to destroy all towers from our new starting point. Repeat this step for each of the remaining $\mathbf{N}$ - 1 towers.

The first step can be done by iterating over each of the towers, calculating the difference between its power and its distance from our starting point. The largest such difference among all the towers is the minimum amount of time Herry must wait before beginning to walk around the circle. Visually, this is the minimum amount we must raise the “line” representing Herry’s power such that it clears all the towers:



Left to right, the differences between power and distance from start are: 1, 4, 1, -5



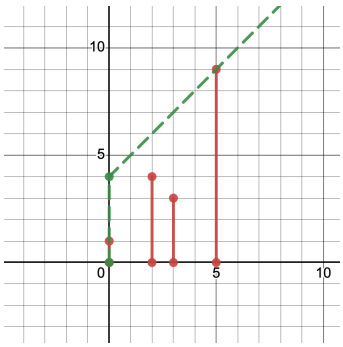
By waiting 4 hours, Herry shifts the green line 4 units up to clear the tower second from left

The second phase involves taking a “step back,” moving back an edge to bring the last tower in the path to the front as our new starting point. This leads to a couple of observations:

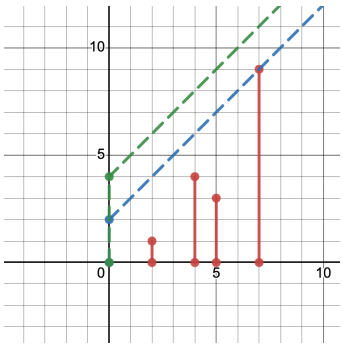
- Since the tower “line segments” move further to the right, our original power line is still able to pass over them—in fact, now it does so by a margin equal to the distance “walked back.” Thus, we can potentially lower our existing power line to save Herry time.
- The tower we bring to the front, however, may force us to raise the line if it is relatively tall (specifically, it is taller than the “discounted” line that passes over all other moved-back segments). In this case, the line must be raised to match its height.

- However, if the tower brought to the front is relatively short, then the discounted line is already raised higher than the height of the new first segment. As such, it doesn't need to be raised any more, and the height can remain the discounted value.
- Since the height of the line corresponds to the amount of time Herry must wait at the beginning, all we have to do is add the necessary walking time to find the total time needed to destroy all towers from our new starting point.

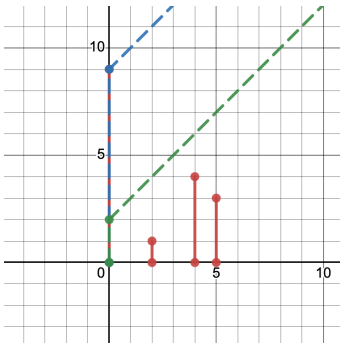
Returning to the example consisting of $\mathbf{N}$ = 4 towers with powers $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$, $\mathbf{P_4}$ = 1, 4, 3, 9 and distances $\mathbf{D_1}$, $\mathbf{D_2}$, $\mathbf{D_3}$, $\mathbf{D_4}$ = 2, 1, 2, 2, taking a step back looks like this:



Before we take a step back and make tower #4 our new starting point

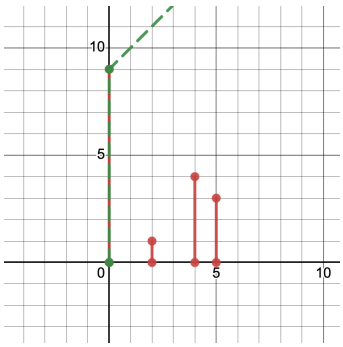


The towers move back $\mathbf{D_4}$ = 2; the line can be lowered by that amount!

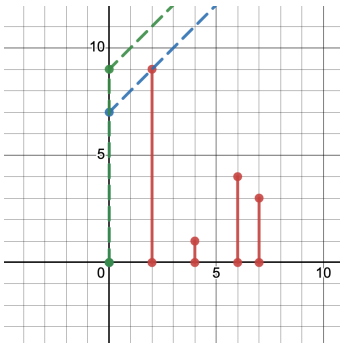


However, bringing the last tower to the front requires the line to be at a height of 9

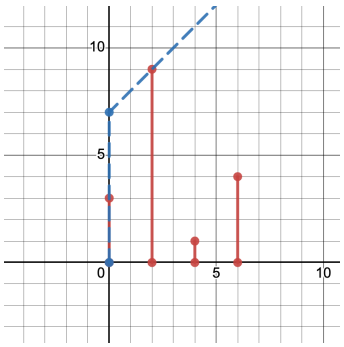
Since Herry needs to be able to destroy the first tower, the new height of the line must be 9; thus, Herry has to wait 9 hours before walking around the circle. Taking another step back looks like this:



Before we take a step back and make tower #3 our new starting point



The towers move back $\mathbf{D_3}$ = 2; the line can be lowered by that amount!



After bringing the last tower to the front, our discounted line is tall enough!

Here, the new starting tower only requires a line of height 3, so our discounted line height of 7 is enough to clear the line segment! As a result, Herry only needs to wait 7 hours before walking around the circle.

Formally, let q denote the minimum amount of time Herry must wait before walking around the circle, starting at tower #i. When Herry takes a step back, two things are true:

1. Herry needs to wait $\mathbf{P_{i-1}}$ hours to destroy the (new) first tower #i - 1.
2. Herry needs to wait at least q - $\mathbf{D_{i-1}}$ hours before walking around the circle to destroy the other towers.

The minimum line height that satisfies both these requirements is the maximum of the two. The distance Herry must walk is the sum of all distances $\mathbf{D} = \mathbf{D_1} + \mathbf{D_2} + \ldots + \mathbf{D_N}$ minus the excluded edge leading up to our new starting point of length $\mathbf{D_{i-2}}$. Adding these values together gives us the total amount of time needed for Herry to destroy all towers from our new starting point, tower #i - 1. This step can be repeated for all remaining towers, with the best such time saved to be outputted at the very end.

The first phase takes $O(\mathbf{N})$ time. The calculations involved in taking a step back can be done in constant time, meaning our $\mathbf{N}$ - 1 steps back cause the second phase to also take $O(\mathbf{N})$ time!